Multi-Objective Graph Heuristic Search for Terrestrial Robot Design

Jie Xu, Andrew Spielberg, Allan Zhao, Daniela Rus, Wojciech Matusik

http://moghs.csail.mit.edu

Abstract— **We present methods for co-designing rigid robots over control and morphology (including discrete topology) over multiple objectives. Previous work has addressed problems in single-objective robot co-design or multi-objective control. However, the joint multi-objective co-design problem is extremely important for generating capable, versatile, algorithmically designed robots. In this work, we present Multi-Objective Graph Heuristic Search, which extends a single-objective graph heuristic search from previous work to enable a highly efficient multi-objective search in a combinatorial design topology space. Core to this approach, we introduce a new universal, multi-objective heuristic function based on graph neural networks that is able to effectively leverage learned information between different task trade-offs. We demonstrate our approach on six combinations of seven terrestrial locomotion and design tasks, including one three-objective example. We compare the captured Pareto fronts across different methods and demonstrate that our multi-objective graph heuristic search quantitatively and qualitatively outperforms other techniques.**

I. Introduction

Most physical tasks in the world, performed by humans or other animals, require being adept at multiple skills. For example, a lizard hunting prey may need to be proficient at climbing trees and running; a duck in migration needs to be able to both fly and swim; a human hurdler must be fast at running along bends and straightaways as well as jumping. If such animals were only capable of single motions, we would not expect them to be very successful.

We similarly should expect diverse adroitness from our robots. Yet, recent successes in algorithms which can co-design robots over morphology and control have been typically catered to singlular task specifications. For example, an algorithm may be able to co-design robots for forward running speed, energy efficient gaits, or climbing rough terrains, but not all three skills at once. In order to computationally develop robots capable of composite tasks rather than single repetitive motions, we require algorithms that simultaneously optimize over collections of requisite skills.

This work presents a method for multi-objective rigid robot co-design over both control and morphology (including discrete topology). Unlike much previous work on multi-objective co-design which only examined continuous parameters, we consider discrete topology as well as continuous control parameters. Because form informs function and vice versa, it is natural that different robot designs will be better

J. Xu, A. Spielberg, A. Zhao, D. Rus, W. Matusik are with the MIT Computer Science And Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts, {jiex,aespielberg}@csail.mit.edu,azhao@mit.edu,{rus,wojciech}@csail.mit.edu

at different tasks. But, as in nature, rarely will a single design be best at all tasks. Thus, our goal is to extract robots with optimal trade-offs across different design objectives; *i.e.* the Pareto set of robot designs for the tasks at hand.

Our method builds upon RoboGrammar [1], a method that proposes a bio-inspired grammar for robot topological design and employs a learning-based morphological search over that design space. Robots are co-designed over topology and control; specifically, it employs a model predictive control (MPC) scheme. Complex grammars like that of RoboGrammar can be very expressive but typically yield large search spaces that are intractable to optimize over *via* naïve methods; the problem only becomes more difficult when multiple objectives are considered. The algorithm we present here is the only proposed method for solving this difficult and important multi-objective, topology/control co-design problem; thus making it novel in both problem scope and solution.

In this work, we contribute: *1)* Multi-objective co-design algorithms for finding Pareto-optimal robot topologies and controllers over challenging objective trade-offs, *2)* Demonstrations on combinations of terrestrial robot locomotion tasks, some with design restrictions, and *3)* Comparisons of our proposed methods benchmarked against baselines, demonstrating the power and importance of our techniques.

II. Related Work

A. Multi-objective Optimization

Multi-objective search algorithms have been successfully deployed in a wide variety of engineering domains, including material design [2], automotive engineering [3], thermodynamics [4], and medicine [5], to name a few. Core to these applications is the development of search algorithms that can retrieve dense Pareto fronts that are close to the ground-truth, with high sample efficiency. Two popular categories of strategies exist. The first is evolutionary algorithms; see [6] for an introduction. Popular methods in this space include NGSA-II [7], NEAT [8], and CMA-ES [9]. These algorithms employ principled heuristics to efficiently trade off exploration of the design space with exploitation of the estimated Pareto front. Contrasted with evolutionary approaches are analytical methods such as [10] or [11], which combine probabilistic search with local, gradient-based optimization for increased efficiency. Particularly popular in this space are scalarization approaches (like [10]), which perform continuous optimizations over sampled weight combinations. Our approach is similar to scalarization methods, but makes significant adap-

tations in order to account for large, grammar-based design spaces with stochastic objectives.

B. Multi-Objective Control Optimization

Multi-objective optimization has been recently applied to robot control problems through the combination of reinforcement learning methods and classical multi-objective optimization techniques. Methods such as [12] and [13] apply scalarizations to reduce these multi-objective searches to single objective problems that can be solved *via* reinforcement learning algorithms. Another category of methods discovers the entire set of control policies to approximate the true Pareto-optimal set, either through preference sampling [14], [15], evolutionary algorithms [16], or universal control policy representation [17], [18]. Compared to those methods, our work focuses on co-design of complex topology and control, and understanding their tight interplay.

C. Robot Co-Design

Robot co-design techniques include evolutionary approaches, analytical methods, and search-based approaches.

Evolutionary approaches apply evolutionary algorithms to iteratively improve robot form and behavior across generations. Early notable work in this space includes [19] and [20], which evolved rigid robot morphologies and controllers to produce agile creatures. It was recently demonstrated that such techniques could be applied to the robust space of neural network controllers [21]. Such techniques have also been applied to soft robots [22], [23], [24], evolving robots over geometry, actuation, and open-loop control. Evolutionary approaches lead to high diversity but poor convergence.

Analytical methods use model gradients in order to inform search. Analytical methods improve search efficiency, but can get stuck in local suboptima. Rigid co-optimization over continuous morphological and control parameters for walking robots has been extensively studied over the last decade [25], [26], [27], [28], [29]. Similar continuous parameter searches have also been applied to soft robots [30], [31].

Search-based methods can both handle combinatorial topology search and still have promising efficiency. In these methods, discrete searches of joint and limb configurations are guided by heuristic functions. These heuristics can be provided *a priori* [32] or, most similar to our work, iteratively learned [1] from evaluated designs.

III. PRELIMINARIES

Our method builds upon RoboGrammar, a grammar-based search method for co-designing robot morphologies and controllers. Choosing a grammar as a search space enables one to search over discrete operations that define robot topology; this provides for much more expressive designs than afforded by purely continuous parameters. We refer the reader to [1] for details regarding the RoboGrammar method and briefly summarize its key components here.

In RoboGrammar, each robot topology design is represented as a directed acyclic graph, where each graph node is corresponding to a physically realizable component and each graph edge is corresponding to a physical link between those parts. RoboGrammar's morphological design space is
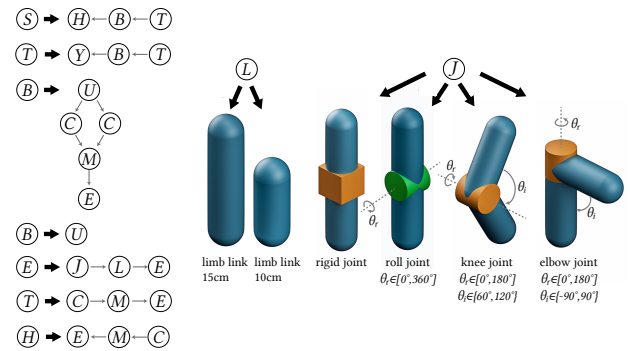

Fig. 1. A graph-based grammar for terrestrial robot proposed by RoboGrammar. **Left:** A subset of the grammar's rules. **Right:** Physical components that can be generated by next-to-terminal symbols. The full grammar can be found in [1].

then defined by a context-free graph grammar, and is constructed in order to promote terrestrially locomoting designs inspired by arthropods. Fig. 1 provides a snapshot of some of the most salient operations it describes. Each instance of the grammar has a graph representation where nodes can correspond to terminal or nonterminal symbols. Starting from a nonterminal symbol "S", the grammar iteratively applies applicable rules to replace a nonterminal symbol in the current topology graph by a subgraph. A terminal design is produced once the graph consists of only terminal symbols. The grammar provides a constrained (*i.e.* valid) but expressive design space. While we use this grammar for these listed advantages, we stress that the search algorithm presented in this paper is general and can be applied to any grammar-defined robot search space.

To search over its combinatorial design space, RoboGrammar proposes Graph Heuristic Search (GHS) algorithm, which is inspired by reinforcement learning. The algorithm alternates a heuristic-function-guided stochastic sampling of robot designs with a learning phase which improves that heuristic function. The heuristic function's goal is to accurately predict the "reward" that a design will provide, including those of partial (non-terminal) designs generated during the design generation process. Despite the search space being combinatorial, the GHS prioritizes the branches that have the best chance of producing well-performing deigns; readers may draw parallels to A* search. This heuristic is defined by a graph neural network architecture which naturally maps design's links to nodes and joints to edges. The heuristic can operate on both complete and partial designs, enabling its use throughout the search.

In order to evaluate sampled designs and generate a control sequence, RoboGrammar employs MPC, specifically a stochastic MPPI scheme based off the POLO algorithm [33]. This stochasticity allows for exploration in the control planning stage and is resilient to poor local minima; however, this comes at the cost of noise in the evaluation procedure. Applying MPC twice to a morphology can lead to different controllers with very different rewards. This stochasticity requires care in the design generation phase, as it means the GHS must (judiciously) revisit designs to see if better controllers are feasible.

We note that although this co-design search may seem

to be just two separate, sequential steps — a morphological design search followed by control generation — it is more accurate to describe it as a three-phase alternating co-optimization process when the heuristic is involved. First, a heuristic is used to "intelligently" sample a design; second, the design is fed to MPC, and a motion is optimized; third, the reward calculated from the MPC-generated trajectory is used to improve the accuracy of the learned heuristic function. This three-phase process is visualized in Fig. 3.

The remainder of this paper is structured as follows. First, we introduce the scalarization approach to multi-objective optimization, and present two approaches to robot co-design which leverage this approach: a simple method based on solving a set of standalone subproblems, and a more sophisticated and efficient Multi-Objective Graph Heuristic Search (MOGHS) that uses GHS to iteratively expand the Pareto front along different sampled directions in objective space, while learning a heuristic function shared across these directions. This shared heuristic is key to making our search efficient. Then, we present experiments demonstrating the efficacy of our methods compared to baselines, and conclude with possible extensions to this work.

IV. METHOD

Let $\mathcal{D}$ define the space of valid designs (morphology and control sequence). We define a multi-objective function $\mathbf{F} : \mathcal{D} \to \mathbb{R}^m$, such that for $d \in \mathcal{D}$, $\mathbf{F}(d) = (f_1(d), f_2(d), ..., f_m(d))$. Our goal is to generate designs which are Pareto optimal in the objective space; in other words, designs which are *non-dominated* by any other discovered design for any objective trade-off. In our maximization problem setting, a design d is said to be Pareto optimal if $\nexists d' \in \mathcal{D}$ s.t. : $\forall i\ f_i(d') \geq f_i(d) \ \land\ \exists i$ s.t. $f_i(d') > f_i(d)$. In layman's terms, a design is considered Pareto optimal if there does not exist another design that is strictly better than this design at all objectives. We call the set of objective values of Pareto optimal designs the *Pareto front*.

In practice, computing the exact Pareto set/front is intractable for most hard problems; thus, we seek an algorithm which can find a Pareto approximation set that is as "good" as possible. We discuss quantitative metrics for evaluating the quality of a Pareto front in Sec. V-B. However, we describe a qualitative way of determining the goodness of Pareto fronts here. Consider weight vectors $\boldsymbol{\omega} \in \mathbb{R}^m$ s.t. $\forall i\ \omega_i \geq 0$ and $\|\boldsymbol{\omega}\|_p = 1$ for some norm p. This can be thought of as the space of rays that sweep out radially in the first orthant. We wish to find a
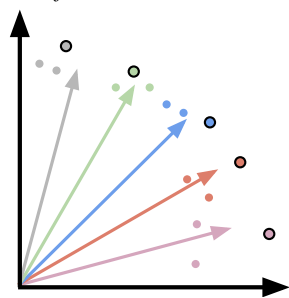
Fig. 2. A cartoon depicting the scalarization method. Weight pairs form rays that project radially outward from the origin. Each circle represents a point that might be found during a single objective optimization using the weights defined by the ray of its color. Circles with black borders are the optimal solutions to the corresponding weights, which form a convex Pareto approximation front.

Pareto set such that, for every valid $\boldsymbol{\omega}$, there exists a point d in the Pareto set for which $\boldsymbol{\omega} \cdot \mathbf{F}(d)$ is large. In other words,
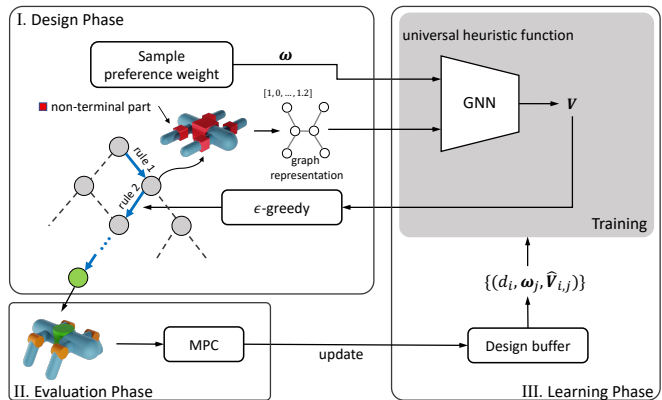

Fig. 3. Overview of the Multi-Objective Graph Heuristic Search (MOGHS). In each episode, the algorithm conducts three phases (similar to GHS). *Design Phase*: A robot design is selected using a learned universal graph heuristic function along with a randomly picked preference weight $\boldsymbol{\omega}$. *Evaluation Phase*: The selected robot design is evaluated by MPC for each objective. *Learning Phase*: All the designs seen so far are leveraged to improve the heuristic.

for each (scalarization direction) ray $\boldsymbol{\omega}$ we want to find points whose objectives are far away from the origin along that ray.

Given this, an obvious strategy for multi-objective optimization arises, termed *scalarization* methods. For a large collection of weight vectors $\{\boldsymbol{\omega}^i\}_{i=1}^n$, a Pareto approximation set can be extracted by solving the set of optimization subproblems, where subproblem $\mathcal{P}_i$ is $\arg\max_{d \in \mathcal{D}} \boldsymbol{\omega}^i \cdot \mathbf{F}(d)$ (Fig. 2). This approach leads to two challenges. The first challenge is to find (as close to) the global maximum of each optimization subproblem. The second challenge is to have an efficient optimization scheme such that a dense set of weight vectors can be optimized.

As a first attempt at solving this problem, we propose the following (naïve) "discrete weights" strategy. Given a budget n, sample a uniformly spaced set $\{\boldsymbol{\omega}^i\}_{i=1}^n$ *a priori*. Then, for each weight vector $\boldsymbol{\omega}^i$, solve the n corresponding $\mathcal{P}_i$ independently, using the approach presented in [1] as a black box with reward $\boldsymbol{\omega}^i \cdot \mathbf{F}$. Such an approach can unfortunately have poor sample efficiency, especially in high-dimensional objective spaces, and treats problems with shared structure as decoupled. Thus, here we propose an alternative algorithm that is more effective at extracting good Pareto fronts.

A. Multi-objective Graph Heuristic Search

The discrete weights strategy has two shortcomings. The first is its decision to fix weights *a priori*. This makes it harder to find Pareto optimal points that do not lie along the sampled scalarization directions. The second shortcoming is the decision to treat each subproblem independently, despite the clear shared structure between the subproblems.

We thus improve the approach presented in RoboGrammar to efficiently search in multi-objective spaces. Our multi-objective optimization strategy makes the following two core changes. First, we sample weight vectors uniformly randomly at each episode of the algorithm both to search over the entire space of weight vectors in a *single* invocation of search algorithm, as well as to have a *dense* set of scalarization directions. Our second core innovation is to

modify our learning model to be *multi-objective* and *universal* (named after its similarity in structure to universal control evaluators [17]), which we describe shortly. This improves search efficiency because the representations learned by the heuristic function can be shared by all weight combinations.

The process of MOGHS is visualized in Fig. 3, and the full algorithm can be found in Algorithm 1. Here, the original GHS is presented in black and teal as in Zhao et al. [1], with our new modifications highlighted in red. We now describe the major changes to the three phases of RoboGrammar's GHS to formulate our MOGHS.

1) Design Phase: In GHS, a designs is sampled in each episode according to a "double ε-greedy" approach in order to balance exploration and exploitation. First, K designs are sampled by ε-greedily choosing the next rule to apply at each step of the generation process, where the greedy choice is chosen by the one with the best heuristic function value. The final design can then be chosen greedily or rejected for a random one at an ε rate. In MOGHS, we instead first sample an $\boldsymbol{\omega}$ vector uniformly at random, a search direction for this episode of the design search. Using this, the greedy selection scheme is generalized to multiple objectives by the linear scalarization $\boldsymbol{\omega} \cdot \mathbf{V}(d, \boldsymbol{\omega})$.

Readers may note that the new greedy selection criterion requires two changes to the structure of the heuristic function. First, the heuristic must accept a weight vector $\boldsymbol{\omega}$ in addition to a design d as input, thus making it a *universal* predictor among all possible $\boldsymbol{\omega}$. Second, it outputs a vector of the predicted value of each objective function when evaluated by MPC, thus making it *multi-objective*. Thus, we design such a heuristic function $\mathbf{V} \colon \mathcal{D} \times \mathbf{R}^m \to \mathbf{R}^m$.

2) Evaluation Phase: A candidate design is evaluated m times, in m separate invocations of MPC. The ith invocation is run optimizing reward (objective) f_i. This returns m different optimized control sequences for the sampled design[1].

3) Learning Phase: Given sampled batch of designs $S_b = \{d_i\}$ and weights $W = \{\omega^j\}$, we first compute the target heuristic value $\hat{\mathbf{V}}_{ij}$ for each design-weight combination. Then we train our universal heuristic function by using Adam [34] to minimize the following loss function:

$$\sum_{i,j} \|\mathbf{V}(d_i, \omega^j) - \hat{\mathbf{V}}_{ij}\|_2^2 \qquad (1)$$

The target heuristic value $\hat{\mathbf{V}}$ is computed by maintaining a Pareto front of rewards for each partial design to keep track of the optimal terminal designs can be induced from it.

B. Universal Multi-Objective Heuristic Function

Our heuristic function is a graph neural network which maps robot morphologies to output vectors of predicted rewards. The morphology is represented as a graph, where each node corresponds to a link of the robot (and whose corresponding feature vectors describe their geometric and inertial properties), and each edge corresponds to a joint, thus encoding the topology. The architecture used is based on the differentiable pooling architecture presented by [35].

---

[1]If f_i does not depend on the robot's motion (*e.g.* the design complexity objective in section V), this process can be skipped for that objective.

We wish for this network to operate on both a robot morphology as well as a weight vector as input; however, this architecture only handles graphs. In order to learn latent representations in the network which include the effects of the objective weights, we include $\boldsymbol{\omega}$ as additional features to each node in the graph. Finally, we modify the final linear operator of the network to output m channels instead of one, thus providing a multi-objective output space of the heuristic.

C. Other Improvements

We include the following further modifications which improve sample efficiency:

1) DAG-Based Target Updates: In [1], when a design is evaluated, its value is propagated up the derivation path to update the target value of the partial designs that generated it. Realizing the design rules actually form a directed acyclic graph (DAG) — each partial or complete design can have many rule sequences that generate it — we now perform the upward propagation procedure even on non-evaluated designs, which provides an opportunity to merge previously visited partial designs up newly discovered branches of the DAG. This simple refinement vastly improves sample efficiency by improving the accuracy of the computed target values.

2) Invalid Design Marking: Although the grammar is designed to avoid invalid designs, they still can occur. Once we know a design is invalid, however, we need not visit it again. We mark all invalid designs as such, and remove them from the pool of candidates to generate. If all designs that could be generated by a partial design are invalid, we further mark that partial design as invalid in an upward propagation scheme similar to the DAG-based update.

D. Implementation

Despite MOGHS requiring many samples over design and control, the algorithm provides many opportunities for parallelization over CPU cores, thus keeping it practical. First, MOGHS samples many designs in parallel, which can be parallelized over many CPU cores. Second, the main bottleneck of MOGHS is the evaluation phase; fortunately, our MPC algorithm is sampling-based; this sampling procedure can also be parallelized over CPU cores to improve efficiency. Finally, the learning procedure can be accelerated by batching MN_W samples in parallel. Running 2000 episodes of MOGHS takes approximately 20 hours on a 64-core Google Cloud machine, and the breakdown for each phase is around 3 hours (using 16 cores) for the design phase, 11 hours (using 64 cores) for the MPC evaluation phase, and 6 hours (using 1 core) for the learning phase.

V. EXPERIMENTS

We compare the Pareto fronts discovered by three algorithms: *1)* a random baseline, in which designs are sampled by randomly selecting rules until a terminal design is generated , *2)* The discrete weights method proposed in IV, which is a discrete version of our MOGHS algorithm, and *3)* our MOGHS algorithm. We use the same total MPC evaluation budget (*i.e.* number of evaluated designs) for all three algorithms. We demonstrate our algorithm on six

Algorithm 1 Multi-Objective Graph Heuristic Search

Inputs: Number of iterations N, number of candidate designs K, Adam optimization steps opt_iter and batch size M, number of sampled weights N_w.

Output: A set of Pareto-Optimal designs P.

Initialize the universal graph neural network $\mathbf{V}_\theta(d, \boldsymbol{\omega})$.

Initialize the Pareto-Optimal design set $P \leftarrow \{\}$.

for episode $j \leftarrow 1$ **to** N **do**
   ▷ **Design Phase:** Generate a candidate design
   Sample a preference weight $\boldsymbol{\omega}$.
   $C \leftarrow \{\}$ ▷ Initialize possible design candidates
   ▷ Sample K designs by ε-greedy approach
   for $k \leftarrow 1$ **to** K **do**
      $d \leftarrow$ initial design graph
      while d has non-terminals **do**
         With probability ε select a random rule a, otherwise select $a = \arg\max_a \boldsymbol{\omega} \cdot \mathbf{V}_\theta(d', \boldsymbol{\omega})$, where d' is the robot design after applying rule a on design d.
         $d \leftarrow d'$
      end while
      Add possible candidate d to C.
   end for
   ▷ Choose one to be the candidate
   With probability ε select a random sampled design d from C, otherwise select $d = \arg\max_{d \in C} \boldsymbol{\omega} \cdot \mathbf{V}_\theta(d, \boldsymbol{\omega})$.
   ▷ **Evaluation Phase:** Compute the rewards for the design
   Run MPC in parallel for each task to evaluate the rewards vector $\vec{r}$ of design d.
   ▷ Update the design Pareto set
   Update P by d and $\vec{r}$.
   ▷ **Learning Phase:** train heuristic value function $\mathbf{V}_\theta$
   for $i \leftarrow 1$ **to** opt_iter **do**
      Sample a minibatch S_b of seen designs (partial or complete) of size M.
      Sample N_w preference weights $W = \{\boldsymbol{\omega}^j\}$.
      Compute target values for each $s \in S_b$ and $\boldsymbol{\omega} \in W$,

$$\hat{\mathbf{V}}(s, \boldsymbol{\omega}) = \arg\max_{d \in \text{descendant}(s)} \boldsymbol{\omega} \cdot \vec{r}(d)$$

      Update $\mathbf{V}_\theta(s, \boldsymbol{\omega})$ one step by Adam with the loss:

$$\sum_{s \in S_b, \boldsymbol{\omega} \in W} \|\mathbf{V}_\theta(s, \boldsymbol{\omega}) - \hat{\mathbf{V}}(s, \boldsymbol{\omega})\|^2$$

   end for
end for

---

combinations of seven tasks, and compare each solution set qualitatively and quantitatively; please see the video for demonstrations of robots along the discovered Pareto fronts.

Flat Terrain Locomotion: In this task, the robot is rewarded for the forward running speed, and we assign additional reward to encourage stability in the forward direction.

Low Power Flat Terrain Locomotion: The same as the Flat Terrain task, except the maximum impulse of the motors is set to 20% of that normally available. This task highlights locomotion in scenarios when power must be conserved.

Wall Terrain Locomotion: Also the same as the Flat Terrain task, however, slalom-like walls are added to the terrain. Successful robots must run forward with the ability to move somewhat laterally to navigate terrain.

Jumping: In this task, the robot must jump as high as possible. The reward is set proportional to the height of the lowest part of the robot. As before, an additional reward is added to discourage the robot from falling over.

Spin-In-Place: This task tests the agility of the robot around the vertical axis. The reward is set proportional to the angular velocity of the robot around the vertical axis.

Design Complexity: The first of two tasks that is purely design-dependent (does not involve control), the reward is set inversely proportional to the number of links in the robot.

Robot Height: The second pure design task, the reward is set proportional to the height of the robot, with a penalty for changes in pitch, promoting tall, upright robots.

A. Experimental Setup

We run each experiment for 2000 episodes. For each task combination, we run each algorithm three times. In comparing metrics in Table I, we compute the metric by averaging over all runs for that algorithm. For metrics that require a reference set, we take the union of all sampled designs of all runs of all algorithms, and compute its Pareto front. Hyperparameters used for the MOGHS algorithm are the same as GHS [1], with the preference weight minibatch size N_W set to 10. For the discrete weights algorithm, we sample 11 uniform weights in the two-objective cases (we did not consider this baseline in the three-objective case).

B. Results

We numerically evaluate the optimized Pareto fronts on three metrics, commonly used in the multi-objective optimization literature [36]: Hypervolume, Generational Distance, and Inverse Generational Distance. We present some Pareto fronts in Fig. 4, along with some selected designs. We encourage the reader to watch the video for more renderings of optimized Pareto fronts, and animations of the designs that populate them.

a) Hypervolume [37]: The hypervolume metric (HV) measures the hypervolume of the polytope defined by the space enclosed by the hyperplanes created by the axes along the first orthant and the hull of the sampled points. As is visually evident, a larger HV is better.

b) Generational Distance [38]: The generational distance (GD) of a Pareto front P is defined as:

$$GD(P) = \frac{1}{|P|}\left(\sum_{\mathbf{p} \in P} \min_{\mathbf{r} \in R} d(\mathbf{p}, \mathbf{r})^p\right)^{\frac{1}{p}} \quad (2)$$

where p is a norm (we choose 1), d is the Euclidean distance function, each $\mathbf{p}^i$ is a point in P, and the set R is a reference set generated by combining results from all optimization experiments. A smaller GD is better.

c) Inverse Generational Distance [39]: The inverse generational distance (IGD) is defined as:

$$IGD(P) = \frac{1}{|R|}\sum_{\mathbf{r} \in R} \min_{\mathbf{p} \in P} d(\mathbf{r}, \mathbf{p}) \quad (3)$$

Again, a smaller IGD is better. Roughly speaking, GD measures the distance of all points on the captured Pareto front to the *best* known Pareto front, while IGD measures the distance of all points on the *best* known Pareto front to the captured Pareto front.

Numerical results are presented in Table I. As can be seen in MOGHS dominates discrete weights in all task combinations across all metrics, often by significant margins.
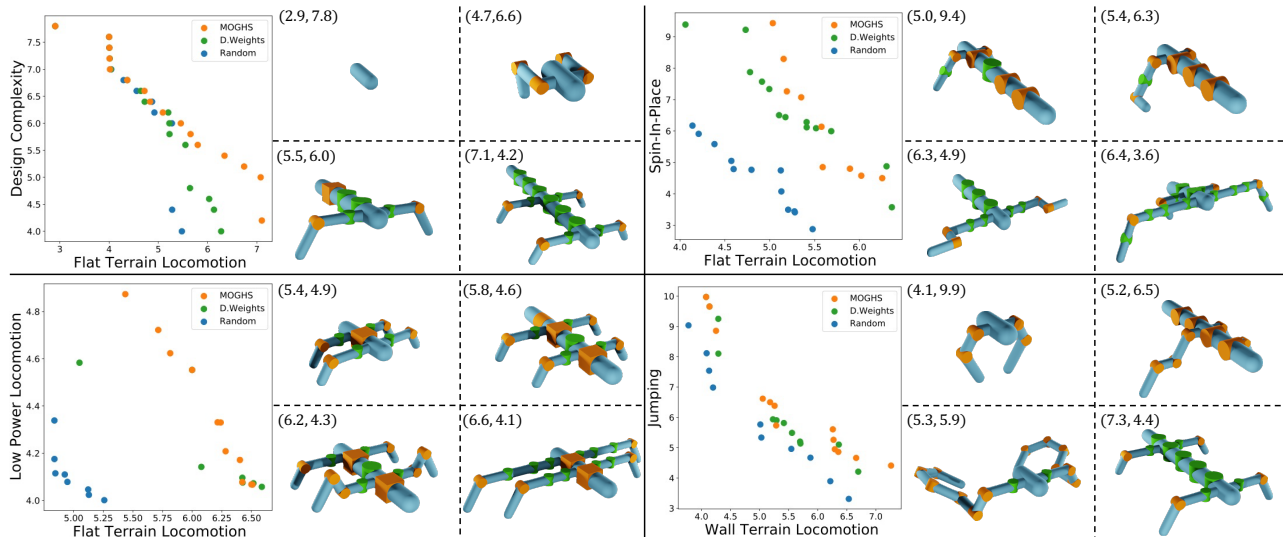
Fig. 4. Pareto front comparison of four of our two-objective experiments, and example designs from the Pareto front. MOGHS produces more diverse and better performing results than discrete weights or the random baseline.

Discrete weights, in turn tends to beat the random baseline, but not as consistently. The consistency and quality of results returned by MOGHS emphasizes the importance of this method. Qualitatively, MOGHS's Pareto fronts, as seen in Fig. 4 tend to yield better performing objective trade-offs than other methods, while maintaining dense coverage.

The morphologies and motions of the designs found on the Pareto fronts of each problem are physically principled, but still interesting and exciting. We consider four of the two-objective trade-offs here. We leave the Design-Height problem, which serves as a benchmarking example, and the complex three-objective Flat-Jump-Spin task, which is better visualized animated, for the video. For example, in the Flat-Design task, a wide spectrum of robots from a single link (simplest design but no motion) to long, complex, fast, walkers are recovered. Along the way, various slower walkers with fewer links are found along the front, with increasingly dynamic motions. The Flat-LowPower task measures robots in various stages of power consumption. In the low power configuration, the robot is unable to balance if the legs are too wide, due to the increase gravitational torque on the torso. This leads to poor forward locomotion. However, longer legs lead to faster strides for the normal power state robots. The trade-off provides a spectrum of robots of varying width and compactness, trading off the importance of being effective at forward locomotion in the two power states. The Flat-Spin trade-off produces a very exciting and surprising result, as faster spinners trade off forward locomotion skill for an ability to spin in a top-like fashion. The fastest spinners possess motions that resemble breakdancing. The Wall-Jump robots meanwhile transition from maneuverable walkers to increasingly frog-like morphology with increased capacity to hop.

VI. CONCLUSION

We have demonstrated methods for co-designing robots over morphology and control over multiple objectives. Our multi-objective graph heuristic search is first of its kind,

TABLE I
A comparison of the three numerical metrics among all three algorithms presented. For each problem, metrics are presented in the following order: HV, GD, IGD. Bolded numbers mean that column's algorithm performed best for that algorithm/problem combination. MOGHS outperforms other methods in all metrics across all problems

PROBLEM		MOGHS	D. WEIGHTS	RANDOM
Design-Height	HV	**64.70**	52.43	51.60
	GD	**0.04**	0.45	0.29
	IGD	**0.28**	1.02	0.96
Flat-Design	HV	**46.30**	42.25	39.11
	GD	**0.10**	0.29	0.31
	IGD	**0.19**	0.41	0.56
Flat-Spin	HV	**49.94**	46.90	29.88
	GD	**0.34**	0.40	1.35
	IGD	**0.37**	0.50	1.79
Flat-LowPower	HV	**29.94**	28.03	22.25
	GD	**0.04**	0.26	0.92
	IGD	**0.08**	0.28	1.07
Wall-Jump	HV	**57.95**	54.09	44.11
	GD	**0.37**	0.73	1.23
	IGD	**0.36**	0.64	1.15
Flat-Jump-Spin	HV	**307.68**	-	166.71
	GD	**0.22**	-	1.09
	IGD	**0.38**	-	1.69

and extracts far superior Pareto fronts with higher efficiency than more naïve methods. The tasks demonstrated, including running, jumping, spinning, and obstacle navigation have direct practical value in real-world terrestrial agile robots.

There remain important avenues for future research. First, all examples demonstrated in this paper were tested in simulation. A study fabricating these designs and demonstrating their physical accuracy would be valuable. Second, all examples presented in this paper were for two or three objective trade-offs. It would be interesting to see if this algorithm would scale to higher objective spaces. Finally, although we have demonstrated our algorithm for robotics, most of our method should be general to any grammar-defined domain, excepting the evaluation (control) procedure and choice of heuristic architecture. Extensions of our algorithm to other domains would demonstrate further value of our approach.

REFERENCES

[1] A. Zhao, J. Xu, M. K. Luković, J. Hughes, A. Spielberg, R. Daniela, and W. Matusik, "Robogrammar: Graph grammar for terrain-optimized robot design," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, 2020.

[2] M. Ashby, "Multi-objective optimization in material design and selection," *Acta materialia*, vol. 48, no. 1, pp. 359–369, 2000.

[3] X. Liao, Q. Li, X. Yang, W. Zhang, and W. Li, "Multiobjective optimization for crash safety design of vehicles using stepwise regression model," *Structural and multidisciplinary optimization*, vol. 35, no. 6, pp. 561–569, 2008.

[4] S. Fettaka, J. Thibault, and Y. Gupta, "Design of shell-and-tube heat exchangers using multiobjective optimization," *International Journal of Heat and Mass Transfer*, vol. 60, pp. 343–354, 2013.

[5] C. A. Nicolaou and N. Brown, "Multi-objective optimization methods in drug design," *Drug Discovery Today: Technologies*, vol. 10, no. 3, pp. e427–e435, 2013.

[6] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[7] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," in *International conference on parallel problem solving from nature*. Springer, 2000, pp. 849–858.

[8] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[9] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *Evolutionary computation*, vol. 15, no. 1, pp. 1–28, 2007.

[10] G. Eichfelder, "An adaptive scalarization method in multiobjective optimization," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1694–1718, 2009.

[11] A. Schulz, H. Wang, E. Grinspun, J. Solomon, and W. Matusik, "Interactive exploration of design trade-offs," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[12] Z. Gábor, Z. Kalmár, and C. Szepesvári, "Multi-criteria reinforcement learning," in *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 1998, pp. 197–205.

[13] S. Mannor and N. Shimkin, "The steering approach for multi-criteria reinforcement learning," in *Advances in Neural Information Processing Systems*, 2002, pp. 1563–1570.

[14] S. Parisi, M. Pirotta, N. Smacchia, L. Bascetta, and M. Restelli, "Policy gradient approaches for multi-objective sequential decision making," in *2014 International Joint Conference on Neural Networks (IJCNN)*, July 2014, pp. 2323–2330.

[15] K. Li, T. Zhang, and R. Wang, "Deep reinforcement learning for multi-objective optimization," *arXiv preprint arXiv:1906.02386*, 2019.

[16] J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik, "Prediction-guided multi-objective reinforcement learning for continuous robot control," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[17] A. Abels, D. Roijers, T. Lenaerts, A. Nowé, and D. Steckelmacher, "Dynamic weights in multi-objective deep reinforcement learning," in *International Conference on Machine Learning*, 2019, pp. 11–20.

[18] R. Yang, X. Sun, and K. Narasimhan, "A generalized algorithm for multi-objective reinforcement learning and policy adaptation," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 14 610–14 621.

[19] K. Sims, "Evolving 3d morphology and behavior by competition," *Artificial life*, vol. 1, no. 4, pp. 353–372, 1994.

[20] G. S. Hornby, H. Lipson, and J. B. Pollack, "Generative representations for the automated design of modular physical robots," *IEEE transactions on Robotics and Automation*, vol. 19, no. 4, pp. 703–719, 2003.

[21] T. Wang, Y. Zhou, S. Fidler, and J. Ba, "Neural graph evolution: Towards efficient automatic robot design," *arXiv preprint arXiv:1906.05370*, 2019.

[22] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, 2013, pp. 167–174.

[23] N. Cheney, J. Clune, and H. Lipson, "Evolved electrophysiological soft robots," in *Artificial Life Conference Proceedings 14*. MIT Press, 2014, pp. 222–229.

[24] F. Corucci, N. Cheney, H. Lipson, C. Laschi, and J. Bongard, "Evolving swimming soft-bodied creatures," in *ALIFE XV, The Fifteenth International Conference on the Synthesis and Simulation of Living Systems, Late Breaking Proceedings*, vol. 6, 2016.

[25] K. Wampler and Z. Popović, "Optimal gait and form for animal locomotion," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3, pp. 1–8, 2009.

[26] A. Spielberg, B. Araki, C. Sung, R. Tedrake, and D. Rus, "Functional co-optimization of articulated robots," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5035–5042.

[27] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Joint optimization of robot design and motion parameters using the implicit function theorem." in *Robotics: Science and systems*, 2017.

[28] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter, "Jointly learning to construct and control agents using deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9798–9805.

[29] M. Geilinger, R. Poranne, R. Desai, B. Thomaszewski, and S. Coros, "Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–12, 2018.

[30] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "Chainqueen: A real-time differentiable physical simulator for soft robotics," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6265–6271.

[31] A. Spielberg, A. Zhao, Y. Hu, T. Du, W. Matusik, and D. Rus, "Learning-in-the-loop optimization: End-to-end control and co-design of soft robots through learned deep latent representations," in *Advances in Neural Information Processing Systems*, 2019, pp. 8284–8294.

[32] S. Ha, S. Coros, A. Alspach, J. M. Bern, J. Kim, and K. Yamane, "Computational design of robotic devices from high-level motion specifications," *IEEE Transactions on Robotics*, vol. 34, no. 5, pp. 1240–1251, 2018.

[33] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," *arXiv preprint arXiv:1811.01848*, 2018.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[35] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Advances in neural information processing systems*, 2018, pp. 4800–4810.

[36] T. Chen, M. Li, and X. Yao, "How to evaluate solutions in pareto-based search-based software engineering? a critical review and methodological guidance," *arXiv preprint arXiv:2002.09040*, 2020.

[37] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[38] D. A. Van Veldhuizen, "Multiobjective evolutionary algorithms: classifications, analyses, and new innovations," AIR FORCE INST OF TECH WRIGHT-PATTERSONAFB OH SCHOOL OF ENGINEERING, Tech. Rep., 1999.

[39] C. A. Coello Coello and M. Reyes Sierra, "A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm," in *MICAI 2004: Advances in Artificial Intelligence*, R. Monroy, G. Arroyo-Figueroa, L. E. Sucar, and H. Sossa, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 688–697.